Vers une réification de l'énergie dans le domaine du logiciel

L'énergie comme ressource de première classe

Jean-Marc Menaud, Adrien Lèbre, Thomas Ledoux, Jacques Noyé
Pierre Cointe, Rémi Douence, Mario Südholt

Équipe ASCOLA (EMNantes-INRIA, LINA)
prénom.nom@emn.fr

Résumé

En quelques années, le problème de la gestion de l'énergie est devenu un enjeu de société. En informatique, les principaux travaux se sont concentrés sur des mécanismes permettant de maîtriser l'énergie au niveau du matériel. Le renforcement du rôle de l'informatique dans notre société (développement des centres de données, prolifération des objets numériques du quotidien) conduit à traiter ces problèmes aussi au niveau du logiciel.

Dans ce papier, nous nous posons la question de la réification de l'énergie comme fut posée en son temps celle de la réification de la mémoire (l'espace) et de l'interpréteur (la machine d'exécution). Le défi est d'abord de sensibiliser l'utilisateur final au problème de la consommation énergétique en visualisant, grâce à des mécanismes d'introspection, sa consommation, à l'image de ce qui se fait aujourd'hui dans le domaine automobile (consommation instantanée d'essence). Il s'agira ensuite de proposer aux développeurs des mécanismes d'intercession leur permettant de contrôler cette consommation énergétique. Ces mécanismes réflexifs devront concerner l'ensemble du cycle de vie du logiciel.

1 L'énergie comme nouvelle préoccupation ?

En 2008, 2% de la production électrique mondiale a été consommée par les centres de données utilisés pour l'hébergement de l'Internet. La problématique est d'importance car, d'un point de vue mondial, et selon l'*Environmental Protection Agency* (EPA), il a été estimé qu'entre 2007 et 2012 les serveurs et les centres de données vont doubler leurs besoins en énergie pour atteindre 100 milliards de kWh. Cette augmentation des besoins est liée à l'accroissement du nombre d'internautes [1], des terminaux d'accès [2], des données disponibles [3], et à l'apparition de nouveaux usages comme le *Cloud Computing*, ou l'Internet des Objets [4]. À ce rythme, dans 25 ans, Internet consommera autant d'énergie que l'humanité tout entière en 2008.

D'un point de vue plus global, l'impact des TIC

1. Principalement dans les pays du BRIC : Brésil, Russie, Inde, Chine.

2. Le nombre d'ordinateurs est estimé à 1 milliard en 2008, 2 milliards en 2013, sans compter le nouveau marché des téléphones intelligents (77 millions vendus sur les trois premiers mois de 2009 soit 14 par seconde).

3. 161 exaoctets de données ont été créées en 2006, soit approximativement 3 millions de fois l'information contenue dans tous les livres jamais écrits.

4. D'ici à quelques années, le nombre d'appareils communiquants pourrait être de plusieurs dizaines de milliards, soit plus que le nombre d'êtres humains.

dans la consommation électrique est de l'ordre de 10 à 15% dans les pays développés. Pour la France, l'estimation de la consommation électrique est comprise entre 55 et 60 TWh par an, soit 13,5% de la consommation française. Au niveau des particuliers, les TIC dans leur globalité représentent 30% [5] de l'électricité consommée. Depuis 10 ans, cette consommation a triplé. Cette progression risque d'augmenter fortement dans les prochaines années avec l'apparition d'équipements toujours plus « énergivores » (écrans LCD, passerelles domestiques, etc.).

Par conséquent, une rupture technologique est essentielle pour relever le défi majeur de la maîtrise énergétique des équipements informatiques. Les fondeurs, constructeurs et installateurs de matériels informatiques ont déjà travaillé sur ce problème. Par exemple, dans les centres de données, l'unité d'accueil de nouvelles machines n'est pas le nombre de serveurs à installer ou leur type, mais la puissance électrique consommée. Si des modèles de consommation d'énergie au niveau matériel sont aujourd'hui disponibles, il n'existe pas l'équivalent au niveau logiciel.

Dans ce papier, nous explorons les possibilités d'intégrer l'énergie en tant que ressource de première classe pour le logiciel, et donc de disposer de tels modèles dans le domaine du logiciel, manipulables par le logiciel. Cette réification permettra de maîtriser l'empreinte énergétique d'une application tout au long de son cycle de vie.

2 L'énergie, nouvelle ressource physique ?

En matière de gestion explicite des ressources physiques et depuis la fin des années 40, l'informatique est passée d'un extrême à l'autre. Pendant longtemps, le programmeur a été comptable de toutes les ressources qu'il utilisait : mémoire, nombre de cycles du processeur, nombre d'accès au disque, etc. Ainsi sous DOS, la mémoire était organisée en blocs de 64 Ko et cela expliquait la taille maximale d'un tableau dans des langages comme Pascal. Pour le langage Lisp, les programmeurs attachaient une grande importance au contrôle de la consommation de la mémoire qui se traduisait dans le nombre de doublets alloués et la taille de la pile. D'où l'utilisation des fonctions de chirurgie (par exemple `nconc` versus `append` pour la concaténation) permettant d'altérer physiquement les listes. « *Ces fonctions sont d'un maniement délicat faisant passer LISP d'un langage d'expressions à un langage de manipulation de pointeurs où les programmes peuvent s'automodifier* » (chapitre 2 de [2]). Avec l'usage des macros, ces fonctions ont donné lieu aux travaux sur la métaprogrammation et la réflexion, dont 3LISP, dans le but initial d'autodécrire des mécanismes de base de l'interpréteur dont la gestion de la mémoire et celle des continuations [4].

À partir des années 80, la mise en œuvre de mécanismes de gestion des ressources, comme les ramasse-miettes, a permis de décharger les développeurs de la contrainte d'une gestion fine des ressources. Ce détachement du programmeur vis-à-vis des ressources a été amplifié par la montée en puissance des capacités des machines suivant la loi de Moore. Les méthodes de programmation et de génie logiciel se sont affranchies (presque) complètement des notions physiques de la machine pour traiter de la rapidité de développement (objets, programmation agile, patrons de conception, usine logicielle) et la facilité de maintenance et d'évolution (architectures à base de composants, d'aspects et de services).

Grâce à ces techniques logicielles, et à la loi de Moore, nous avons pu construire des applications complexes à forte fonctionnalité ajoutée. Mais, alors que la demande continue à croître, il est impératif d'enrayer l'envolée de la consommation énergétique associée. On va donc rapidement se retrouver dans la situation des programmeurs des années 70, avec des contraintes énergétiques plutôt que des contraintes sur la consommation de la mémoire. Peut-on appliquer à l'énergie les mêmes stratégies que celles utilisées pour gérer la consommation de la mémoire ? Contrairement à la mémoire, la consommation énergétique propre d'une application n'est pas directement accessible. Le premier verrou à lever consiste à réifier cette ressource particulière.

5. Étude REMODECE 2007, ADEME, électricité spécifique consommée.

3 Comment réifier l'énergie ?

Pour réifier l'énergie, il faut dans un premier temps la quantifier, donc disposer d'outils logiciels ou matériels permettant de mesurer la consommation électrique d'un quantum d'exécution (instruction de la machine, pseudo-code, exécution d'un service, suivant la granularité choisie). Cependant mesurer la consommation énergétique d'un quantum d'exécution complexe, comme un service, est une tâche ardue. En effet, l'énergie consommée par un quantum d'exécution est la somme des énergies consommées par les ressources physiques utilisées par ce quantum d'exécution (CPU, mémoire, disque, réseau, etc.).

Dans un deuxième temps, il est nécessaire de construire un modèle théorique de la consommation énergétique qui permette d'attacher des quanta d'énergie aux quanta d'exécution. Pour être utilisable, ce modèle doit être compositionnel et réaliste, c'est-à-dire qu'il doit être capable de prédire avec suffisamment de précision la consommation d'un programme par composition simultanée des quanta d'exécution et des quanta d'énergie. Ces besoins peuvent conduire à choisir des quanta de grain assez gros. Travailler à un niveau de pseudo-code semble prometteur [3]. Une première difficulté peut être, suivant le modèle de programmation, la prise en compte de la composition parallèle de programmes. Certains modèles de programmation, comme le modèle de programmation synchrone, peuvent être plus favorable que d'autres. Une deuxième difficulté est, dans un contexte distribué, de prendre en compte les coûts de communication et plus généralement d'infrastructure (assurant la disponibilité des services).

Cette quantification énergétique des applications sera utilisée à la fois statiquement par le programmeur et le compilateur, assistés de systèmes d'analyse statique quantitative (voir, par exemple, [5]), et dynamiquement par le système d'exécution.

Réifier l'énergie au niveau langage permettra à un programmeur d'introspecter son code sous l'angle énergétique, et de le modifier en changeant ses fonctionnalités ou son implémentation, par exemple en développant des algorithmes de type *anytime* [6]. Les algorithmes *anytime* sont des algorithmes dont la qualité des résultats augmente avec le temps de calcul. Ce type d'algorithme permet de contrôler l'énergie affectée à un calcul en interrompant ce calcul lorsque son quota est épuisé.

Réifier l'énergie au niveau du système d'exécution permettra d'attribuer l'énergie disponible aux applications et d'arbitrer ces allocations en cas de pénurie (énergétique) comme cela est fait pour d'autres ressources [1]. Des stratégies d'allocations spécifiques à la gestion énergétique seront probablement à développer.

4 Conclusion

L'informatique au sens large a et aura un impact significatif sur la consommation électrique mondiale. La poursuite de son développement passe par une prise en charge généralisée de la contrainte énergétique au-delà des domaines habituels du matériel et de certains secteurs de l'informatique embarquée. Pour ce faire, il est nécessaire de réifier l'énergie en faisant de celle-ci une ressource accessible statiquement et dynamiquement au niveau de chaque composant applicatif. Il sera alors possible, dans un premier temps, de sensibiliser de manière efficace à la fois les utilisateurs et les programmeurs finaux aux coûts énergétiques et, dans un deuxième temps, de fournir des concepts [6] et des outils de développement ainsi que des systèmes d'exécution susceptibles de significativement réduire ces coûts.

Références

[1] Luc Morau and Christian Queinnec. Resource aware programming. *ACM Transactions on Programming Languages and Systems*, 27(3) :441–476, May 2005.

[2] Christian Queinnec. *LISP Mode d'emploi.* Eyrolles, 1984.

[3] Chiyoung Seo, Sam Malek, and Nenad Medvidovic. Component-Level Energy Consumption Esti-

6. On peut par exemple penser au développement, dans la continuité des algorithmes *anytime*, d'une algorithmique ≪ verte ≫.

mation for Distributed Java-Based Software Systems. In *Component-Based Software Engineering*, pages 97–113. Springer-Verlag, 2008.

[4] Brian Cantwel Smith. Reflection and semantics in LISP. Technical report, Palo Alto Research Center, 1984.

[5] Pascal Sotin, David Cachera, and Thomas Jensen. Quantitative static analysis over semirings : Analysing cache behaviour for Java Card. In Alessandra Di Pierro and Herbert Wiklicky, editors, *Quantitative Aspects of Programming Languages (QAPL '06)*, 2006.

[6] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3) :73–83, 1996.